

VARIAN 620/L QUICK REFERENCE CARD

DAS Assembler Language (v620asm)

Source Formats

Fixed (card)

Label in column 1; then the operation, the variable field, and a comment, separated by blanks. Operation followed by 8+ blanks: no variable field, rest is comment.

Free (paper tape; BLD II's own format)

LABEL, OPERATION, VARIABLE COMMENT

- No label: statement starts with a comma.
- Operation ended by a blank, not a comma: no variable field.

Both formats may mix in one file. The variable field ends at the first blank outside a character constant. A line whose first nonblank character is * is a comment. * right after the operation (LDA*) means indirect addressing. Tabs expand to 8-column stops.

Symbols & Numbers

- **Symbols:** start with a letter, \$, or #, then letters/digits/\$/#. Significant to 6 characters (DAS MR, the default) or 4 (-das4a: DAS 4A/8A).
- **Numbers:** decimal, unless the first digit is 0, which makes it octal (12 is twelve, 012 is ten).

Operators & Expressions

* and / before + and -; otherwise left to right. Parentheses group. 16-bit arithmetic.

* as a term is the location of the current statement's own first word, even in a two-word instruction's second word. In a DATA/PZE/MZE list, it's each item's own word — so DATA (**1)*, (**1)* makes each word point to the next.

Constants

Character

'AB' — Varian 8-bit ASCII, bit 7 set ('A' is 0301). In an expression, a constant's value is its *last two* characters, right justified. Two primes together (") are one literal prime.

Address

(expr) is a 15-bit address; (expr)* is an indirect address (bit 15 set). Valid in DATA, as CALL arguments, as a jump/SEN address, and as an immediate value (LDAI (TBL)).

Literal

=expr in an address field: the address of a pool word holding expr. The pool starts at END's location, or where LTOR BEGI puts it; identical expression *text* shares one word.

Floating Point

) [sign] int . frac [E] [sign] exp, e.g.)5.5,)-4.+20,) .1E-12. A digit and a decimal point are required; E is optional. Valid in DATA, PZE, MZE; each makes two words.

Word 1: sign (bit 15), exponent (bits 14-7, excess-0200), fraction's high 7 bits. Word 2: fraction's low 15 bits. Normalized 1/2 to 1 (1.0 = 040300 000000). Negative: complement the whole first word. Zero: two zero words. No double-precision form — DAS gives no way to ask for it.

Instruction Operand Forms

Form

Instructions

expr	Immediates (LDAI...), jumps/marks/executes (* or (expr)* for indirect) JIF, JIFM, XIF
cond, addr	SEN
code, addr	IME, OME (address defaults to 0)
device[, addr]	EXC/SEL, EXC2/SEL2, INA INB
field value	INAB CIA CIB CIAB OAR OBR OAB, MERG/ZERO/INCR/COMPL/DECR
0-31	The shifts
expr[, idx[, bits]]	Extended (LDAE...): idx 1=X, 2=B, 7/none=direct/indirect; bits ORs into word 1 (postindexing)

Addressing Mode Choice (single-word)

v620asm picks the mode automatically, as DAS does:

1. address < 2048: direct;
2. address 1-512 past the instruction: relative;
3. anything else (including behind the instruction): indirect, through a pointer the assembler generates in the first 512 words.

expr, 1 / expr, 2 index by X / B. LDA* expr and LDA (expr)* are indirect (directly if the address is below 512, otherwise via a generated pointer); that pointer is itself indirect, so the chain continues through the named word.

Instruction Set

Every 620/L mnemonic is accepted (both LDA and LOA spellings). DAS MR's 620/f additions — the "not" conditions (JOFN, JANZ...), BT, SRE, JSR, IJMP, TSA — assemble too, with DAS MR's codes, but v620emu treats them as undefined on a 620/L. See the machine quick-reference card for the full opcode table.

VARIAN 620/L QUICK REFERENCE CARD

Directives, Macros & Invoking v620asm

Directives

Directive	Effect
ORG <i>expr</i>	Set the location counter; label gets the new value
LOC <i>expr</i>	Assemble as if at <i>expr</i> , load where the counter is; program relocates it before running. ORG ends it
<i>n</i> BEGI <i>expr</i>	Create/preset location counter <i>n</i> without switching to it
USE <i>name</i>	Switch counters: blank, SYOR, COMN, a BEGI name, or PREV
<i>s</i> EQU <i>expr</i>	Define symbol <i>s</i>
<i>s</i> SET <i>expr</i>	Define symbol <i>s</i> ; may redefine
<i>s</i> MAX/MIN <i>list</i>	Largest/smallest value
LTOR/IAOR	Where the literal pool / indirect pointers go, before either is used
BEGI <i>addr</i>	
DUP <i>n</i> [, <i>m</i>]	Repeat the next <i>m</i> statements (1-3) <i>n</i> times (up to 32767); not nestable
DATA <i>list</i>	Store words: expressions, character/address constants
PZE/MZE <i>list</i>	DATA with sign bit cleared / set
BSS <i>n</i>	Reserve <i>n</i> words; label gets the first address
BES <i>n</i>	Reserve <i>n</i> words; label gets the address before the next word
ENTR	A subroutine entry word of zero
CALL	JMPM <i>sub</i> , then args as DATA
<i>sub</i> , <i>args</i>	
RETU* <i>expr</i>	JMP* <i>expr</i>
<i>s</i> OPSY <i>mnem</i>	Another name for a mnemonic
<i>s</i> FORM <i>widths</i>	Define a bit-field layout; <i>s</i> values then packs one word
IFT/IFF	Assemble the next statement only if
<i>a</i> [, <i>b</i> [, <i>c</i>]]	$a \neq 0$, $a \neq b$, or $a < b \leq c$ (IFF: only if not)
GOTO <i>sym</i>	Skip statements to the one labeled <i>sym</i> (trailing comma: don't list them)
<i>s</i> CONT/NULL	GOTO targets (NULL also defines <i>s</i>)
<i>n</i> MAC...EMAC	Define macro <i>n</i>
INCL 'file'	Assemble another file here; a relative name is relative to the including file's directory; nests 8 deep
END [<i>expr</i>]	End of source; <i>expr</i> is the execution address
LIST/NLIS/ SMRY/DETL/ EJEC/SPAC	Listing control
PUNC/NPUN/ READ/MORE	Accepted and ignored

Location Counters

Three to start: blank (in use initially, starts at 0), SYOR (at 0), COMN (at 02000). BEGI creates up to 8 more. Each keeps its own location (and LOC offset) while another is in use. The listing shows assembly addresses; the tape and `-octal` show load addresses.

Macros

Definition:

NAME MAC
...body, using P(1), P(2), ... for args...
EMAC

Invocation (looks like an instruction):

NAME A1,A2,...

P(*n*) in the body is replaced by the *n*th argument at the call site.

FORM / Field Packing

s FORM *w*₁,*w*₂,... declares *s* as a template of fields (bit widths, most-significant first). A later line *s* *v*₁,*v*₂,... packs those values, msb-first, into one word.

Invoking v620asm

	v620asm [options] SOURCE
Option	Effect
-o, -output= <i>P</i>	Object tape (default: <i>SOURCE</i> with .ptp)
-b, -bootstrap	Punch a bootstrap-format tape instead
-l, -listing= <i>P</i>	Listing (default: <i>SOURCE</i> with .lst)
-octal= <i>P</i>	Also write address word lines
-n, -check	Check the source; write nothing
-4, -das4a	4 significant symbol characters (default: 6)
-alpha=left right	One-character DATA constant justification (default left)
-pointers=ADDR	Where indirect pointers go (octal, below 01000; default 0200)
-v, -verbose	Report what was written

Errors: v620asm: FILE:LINE: error: MSG, also in the listing. Any error: no tape written (listing always is). One warning: assembling over a word already generated (as ORG *-1 can) — allowed; the later word replaces the earlier.

The Listing

Each line: address, word, and a mode letter (A code, L literal pool, I indirect pointer), then the source line number and statement. EQU/SET show the value; ORG/BSS the address. A statement's further words follow on their own lines. At the end: the literal pool, the pointers, and the symbol table.

Output

Object tape: words in generation order; records break on any address gap, up to 64 words/record. END's operand becomes the execution record. v620emu's load, and BLD II running on it, both read it.

-bootstrap: every word from the lowest to highest address assembled, in the handbook bootstrap loader's format; that loader stops at a zero word, so v620asm warns about any zero or unassembled location in range.

Not Implemented

Relocatable/linked programs: NAME, EXT, the COMN directive, and FORT are errors. No surviving 620 software needs them, so there's nothing to check an implementation against; see docs/MANUAL.md.